KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

A pure proactive scheduling algorithm for multiple earth observation satellites under uncertainties of clouds

Wang J, Demeulemeester E, Qiu D.

KBI_1419

A pure proactive scheduling algorithm for multiple earth observation satellites under uncertainties of clouds

Jianjiang Wang[a,b], Erik Demeulemeester[b], Dishan Qiu[a]

[a] *Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China*
[b] *Research Center for Operations Management, Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Belgium*
Email: jianjiangwang.nudt@gmail.com, erik.demeulemeester@kuleuven.be, ds_qiu@sina.com

Abstract

Most earth observation satellites (EOSs) are equipped with optical sensors, which cannot see through the clouds. Hence, observations are significantly affected and blocked by clouds. In this work, with the inspiration of the notion of a forbidden sequence, we propose a novel assignment formulation for EOS scheduling. Considering the uncertainties of clouds, we formulate the cloud coverage for observations as stochastic events, and extend the assignment formulation to a chance constraint programming (CCP) model. To solve the problem, we suggest a sample approximation (SA) method, which transforms the CCP model into an integer linear programming (ILP) model. Subsequently, a branch and cut (B&C) algorithm based on lazy constraint generation is developed to solve the ILP model. Finally, we conduct a lot of simulation experiments to verify the effectiveness and efficiency of our proposed formulation and algorithm.

Keywords: earth observation satellites, uncertainties of clouds, proactive scheduling, chance constraint programming, sample approximation, branch and cut

1. Introduction

Earth observation satellites (EOSs) are the platforms equipped with sensors that orbit the earth to take photographs of special areas at the request of users [8, 10]. Because of some unique advantages, e.g. an expansive coverage area, long-term surveillance, a high frequency of repeated observations, accurate and effective information access and unlimited airspace borders, EOSs have been extensively employed in earth resources exploration, nature disaster surveillance, urban planning, crop monitoring, etc. With the development of space science and technology, the number of satellites increases continuously. However, satellites are still limited and expensive due to the explosively increased applications. Hence, scheduling plays a nontrivial role in obtaining high observation effective-

ness and efficiency of EOSs, which is to allocate the submitted tasks to EOSs, making the schedule satisfy operational constraints.

Different from traditional scheduling problems, such as the job shop problem, the parallel machine scheduling and project scheduling, EOS scheduling has some particular characteristics:

- Due to the fact that EOSs orbit the earth, tasks can only be observed in the visible scopes of satellites, which means that task observation has specified time window requirements.
- Between two consecutive tasks, the satellite requires doing some operations for transformation, including sensor shutdown, slewing, attitude stability and startup. Hence, it requires sufficient setup time. Besides, because the slewing angle corresponds with the observation angles of the consecutive tasks, the setup time is not only related to the satellite, but also to the positions of the two tasks.
- Memory and energy consumptions cannot exceed the respective capacities of the satellite. Especially energy will not only be consumed for observation, but also for sensor slewing. Hence, similarly to setup time, energy consumption is not only related to the satellite, but also to the scheduled task sequence, which is difficult for modeling and solving.

Up to now, a great number of studies focusing on EOS scheduling have been proposed, in which EOS scheduling was formulated and solved in different ways:

Mathematical programming: Without considering memory and energy constraints, Benoist et al. [3], Habet et al. [19, 20, 21] and Lemaître et al. [28] developed general mathematical programming models for EOS scheduling. Liao et al. [31, 32], Lin et al. [33, 34, 35, 36] and Marinelli et al. [39] proposed the time-indexed formulation of EOS scheduling, and established integer programming models. In addition, integer programming models are also constructed on the basis of a "flow variable" formulation [7, 8, 15, 16]. Hall et al. [22] formulated the problem as a longest path problem with time windows, and suggested an integer linear programming model.

Constraint satisfaction problem: Lemaître et al. [27] and Verfaillie et al. [48] formulated EOS scheduling as constraint satisfaction problems. Agnèse et al. [1], Bensana et al. [4] and Verfaillie et al. [49] proposed valued constraint satisfaction problem (VCSP) formulations for SPOT-5 satellite scheduling, without considering energy constraints.

Knapsack problem: Vasquez et al. [46, 47] and Wolfe et al. [55] formulated EOS scheduling as 0-1 knapsack problems.

Graph-based formulation: Gabrel et al. [13, 14] adopted a directed acyclic graph (DAG) model to describe the satellite scheduling problem. Besides, Sarkheyli et al. [43] and Zufferey et al. [57] modeled EOS scheduling as graph coloring problems.

Besides, Frank et al. [12] adopted the Constraint-Base Interval (CBI) language to describe the problem.

In addition, the solution approaches for EOS scheduling can be classified into the following categories.

Exact algorithms: Agnèse et al. [1] and Bensana et al. [4] proposed depth-first branch and bound algorithms for SPOT-5 satellite scheduling. Also, Benoist et al. [3], Bensana et al. [4] and Verfaillie et al. [49] suggested Russian Doll search algorithms, which are based on branch and bound but replace one search by n successive searches on nested subproblems, using the results of each search when solving larger subproblems, to improve the lower bound on the global valuation of any partial assignment. Besides, Gabrel et al. [14], Hall et al. [22] and Lemaître et al. [28] developed dynamic programming methods to get the optimal solutions of EOS scheduling problems.

Metaheuristics: A large number of metaheuristics were proposed for EOS scheduling, which primarily contain tabu search algorithms [4, 8, 10, 34, 36, 46, 57], genetic algorithms [29, 44, 45, 55], ant colony algorithms [30, 50, 56], local search algorithms [27, 28, 48] and simulated annealing algorithms [17, 18].

Heuristics: Agnèse et al. [1], Bensana et al. [4] and Lemaître et al. [28] proposed greedy algorithms to get feasible solutions for EOS scheduling problems. On the basis of heuristic rules, Bianchessi et al. [6, 9], Hall et al. [22], Wang et al. [51, 52, 53] and Wolfe et al. [55] developed constructive algorithms, which can solve the problem efficiently, without guaranteeing the optimality of the solutions. Bianchessi et al. [7], Lin et al. [33, 36] and Marinelli et al. [39] adopted lagrangian relaxation heuristics to solve the problems, obtaining close-to-optimal solutions.

Practically, EOS observations are extremely affected and blocked by clouds, because most EOSs are equipped with optical sensors that cannot see through clouds [17, 18]. According to statistics [24], currently about 60% of the observations are covered by clouds, which will result in useless observations. Hence, cloud coverage is a nontrivial issue for EOS scheduling, which cannot be ignored. Unfortunately, to the best of our knowledge, among all the previous studies, only a few have considered the impact of clouds. Lin et al. [33, 34, 35, 36] formulated the coverage of clouds as a set of covered time windows, and forbade the tasks to be observed in the covered time windows of scheduling. In practice, the drawback and infeasibility of Lin's approach is that there exist a lot of uncertainties of clouds, which are always changing over time [5, 27] and it is impossible to be forecasted exactly, so decision makers cannot get the deterministic information of cloud coverage before scheduling. Liao et al. [31, 32] considered the uncertainties of clouds, formulated the cloud coverage for each observation window as a stochastic event, and established a model with the objective of maximizing the weighted sum of a function of the profits and the expected number of accomplished tasks.

In this study, we firstly propose a novel assignment formulation of EOS scheduling, in which the energy constraints are formulated as forbidden sequences (this notion is based on the notion of a forbidden set [25, 26] which was used in resource-constrained project scheduling). Considering the uncertainties of clouds, we formulate the cloud coverage for each time window of observation as a stochastic event, and extend the assignment formulation to a chance con-

Table 1: Notations

T	set of tasks, $T = \{1, ..., n\}$
i, j	task index, $i, j \in T \cup \{0, n+1\}$, in which $0, n+1$ are dummy tasks
ω_i	profit of task i, $i \in T$
O	set of orbits, $O = \{1, ..., m\}$
k	orbit index, $k \in O$
b_{ik}	$b_{ik} = 1$ if orbit k is available for the observation of task i, otherwise $b_{ik} = 0$, $i \in T, k \in O$
M_k, E_k	memory capacity and energy capacity of orbit k, $k \in O$
m_k, e_k	memory and energy consumption for each unit time of observation of orbit k, $k \in O$
$[ws_{ik}, we_{ik}]$	time window of observation of task i on orbit k, $i \in T, k \in O$
st_{ij}^k	setup time between task i and task j on orbit k, $i, j \in T, k \in O$
ρ_{ij}^k	energy consumption for slewing between task i and task j on orbit k, $i, j \in T, k \in O$
$\tilde{\lambda}_{ik}$	binary stochastic variable, $\tilde{\lambda} = 1$ denotes that task i can be successfully observed on orbit k, otherwise $\tilde{\lambda} = 0$, $i \in T, k \in O$
p_{ik}	probability that task i will be successfully observed on orbit k, $i \in T, k \in O$
W	sample, a set of scenarios, $W = \{1, ..., \|W\|\}$, where $\|W\|$ is the sample size
w_l	a scenario, $w_l \in W$

straint programming (CCP) model. The sample approximation (SA) method is applied to transform the CCP problem into an integer linear programming (ILP) problem, say the SA problem. With respect to the characteristics of the SA problem, a branch and cut (B&C) algorithm based on lazy constraint generation is designed. Afterwards, a large number of experiments by simulation are conducted to verify the effectiveness and efficiency of the sample approximation and the B&C algorithm.

The remainder of this paper is organized as follows. In the next section we provide some definitions and a formal problem description. Subsequently, Section 3 proposes a novel assignment formulation for EOS scheudling, and then extends the formulation to a chance constraint programming model. In Section 4, we present an approach to solve the problem. Numerical results of our approach are presented in Section 5. The last section offers conclusions and directions for future research.

2. Problem description

In EOS scheduling, users generally submit two types of requests: (1) a target, i.e., a circle with limited dimension, or (2) a polygon which may cover a wide geographical area. Due to its large size, a polygon usually is failed to be observed

in a single orbit and therefore partitioned into multiple strips [8, 10, 51]. In order to facilitate the description, a target can be seen as a single strip. Hence, the tasks in this work are corresponding to the strips that require being observed.

In the previous studies, scholars usually formulate the satellites as the resources, and a task will have at most one observation window on each resource. However, if the scheduling horizon is long enough, a satellite will orbit the earth for multiple orbits and pass over a strip for multiple times. Hence, the observation windows for a task on each satellite will not be unique [51, 52], which makes the problem difficult for modeling and solving. To handle the difficulties, we formulate the orbits of the satellites as the resources. Hence, there will be at most one observation window for each task on each resource, regardless of the length of the scheduling horizon.

Some notations of this study are summarized in Table 1. Let T be the set of tasks (strips) submitted by users and let O be the set of orbits within the scheduling horizon. With each task $i \in T$ is associated a profit ω_i. Each orbit $k \in O$ is associated with a memory capacity M_k, an energy capacity E_k, a memory consumption for each unit of observation time m_k and an energy consumption for each unit of observation time e_k. Let $b_{ik} = 1$ denote that task i can be observed on orbit k, otherwise, $b_{ik} = 0$. $[ws_{ik}, we_{ik}]$ denotes the time window for task i on orbit k, and θ_{ik} denotes the slewing angle. Many of these notions are illustrated in Figure 1. In this work, we only consider non-agile satellites, which have the maneuverability of rolling (slewing), without the maneuverability of pitching. Hence, the time windows for observations are fixed without flexibility, such that the start and finish time of task i on orbit k will be fixed as $[ws_{ik}, we_{ik}]$, and the duration will be $we_{ik} - ws_{ik}$.

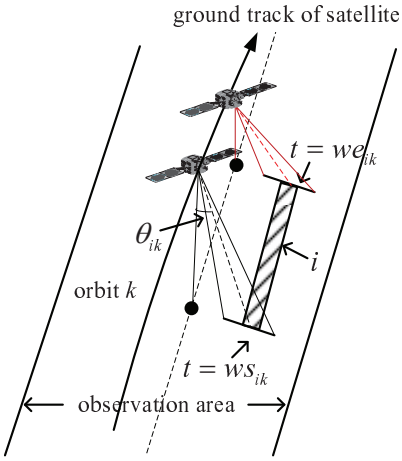

Figure 1: Time window for observation

After observing a task, the satellite requires a sequence of transformation operations to observe the next one, which are sensor shutdown→ slewing→ at-

titude stability→ startup. Hence, there should be sufficient setup time between two consecutive tasks, and the required setup time can be computed by the following formula:

$$st_{ij}^{k} = sd_k + |\theta_{ik} - \theta_{jk}|/s_k + as_k + su_k,$$

where st_{ij}^{k} is the setup time between task i and task j on orbit k, and sd_k, as_k and su_k are the times of sensor shutdown, attitude stability and startup on orbit k, respectively. Besides, s_k is the slewing velocity of orbit k, and θ_{ik} and θ_{jk} are the slewing angles of tasks i and j on orbit k, respectively.

For observing task i on orbit k, the memory consumption can be computed by $(we_{ik} - ws_{ik})m_k$. Different from memory, energy will not only be consumed by observation, but also by sensor slewing. The energy consumption for observing task i on orbit k is $(we_{ik} - ws_{ik})e_k$. Let ρ_{ij}^{k} denote the energy consumption of slewing between consecutive tasks i and j on orbit k, which can be calculated by the formula below:

$$\rho_{ij}^{k} = |\theta_{ik} - \theta_{jk}|\pi_k,$$

where π_k is the energy consumption for each unit slewing angle on orbit k.

Due to the impact and coverage of clouds, some observations will fail. For a single task and the observations on different orbits, namely different time windows, the impact of clouds will be different, which is illustrated in Figure 2. If the satellite observes task i on orbit k, the observation will be blocked by clouds. However, if the satellite observes from orbit $k^{'}$, it will be successful. In addition, for an orbit, namely a single time window, the impact of clouds will not change due to the shortness of the time window, normally a few seconds.

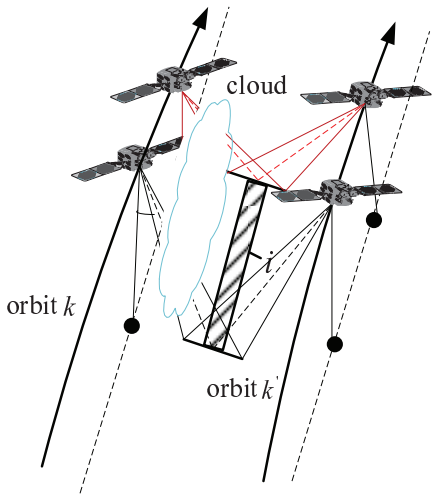

Figure 2: Impact of clouds for different observations

Considering the uncertainties of clouds, we formulate the cloud blocks for observations as stochastic events, denoted by 0-1 stochastic variables $\tilde{\lambda}_{ik}, i \in T, k \in O$. $\tilde{\lambda}_{ik} = 1$ if the observation of task i on orbit k is not blocked by clouds, namely task i can be successfully observed, otherwise $\tilde{\lambda}_{ik} = 0$. Let p_{ik} denote the probability for a successful observation of task i on orbit k, i.e., no cloud block, thus we can obtain $p\{\tilde{\lambda}_{ik} = 1\} = p_{ik}$ and $p\{\tilde{\lambda}_{ik} = 0\} = 1 - p_{ik}$.

3. Mathematical formulations

The mathematical programming models for EOS scheduling can be divided into three categories: time-indexed formulations [31, 32, 33, 34, 35, 36], assignment formulations [4, 56] and flow formulations [7, 8, 15, 16]. At present, flow formulations are applied most widely and get most focus, because the other two have their respective disadvantages. The drawback of time-indexed formulation is that it contains a large number of variables and a large number of constraints, especially for a long scheduling horizon, which makes the problem inefficient to solve [54]. In an assignment formulation it is difficult to model some sequence-based constraints, such as setup time and energy constraints, which usually make the formulation non-linear. In addition, a flow formulation is thought to be better than an assignment formulation, because it has a tighter linear relaxation based upper bound [16]. However, in this study, we propose a novel assignment formulation, which can model the setup time and the energy constraints linearly and be solved efficiently.

3.1. A flow formulation of deterministic EOS scheduling

Before the proposed novel assignment formulation, we firstly describe a flow formulation, which is similar with [15]. In this formulation, we use binary variables $x_{ij}^k \in \{0, 1\}$ $(i, j \in T \cup \{0, n+1\}, k \in O)$, in which $T = \{1, ..., n\}$ is the set of real tasks, and $0, n+1$ are dummy tasks for starting and terminating, repectively. $x_{ij}^k = 1$ if both tasks i, j are scheduled on orbit k, and task i is the immediate predecessor of task j; otherwise $x_{ij}^k = 0$. The integer programming formulation is given below:

$$max \sum_{i \in T} \sum_{\substack{j \in T \cup \{n+1\} \\ j \neq i}} \sum_{k \in O} \omega_i \cdot x_{ij}^k \tag{1}$$

subject to

$$\sum_{\substack{j \in T \cup \{n+1\} \\ j \neq i}} \sum_{k \in O} x_{ij}^k \leq 1, \forall i \in T \tag{2}$$

$$\sum_{\substack{j \in T \cup \{n+1\} \\ j \neq i}} x_{ij}^k = \sum_{\substack{j \in T \cup \{0\} \\ j \neq i}} x_{ji}^k, \ \forall i \in T, k \in O \tag{3}$$

$$\sum_{\substack{j \in T \cup \{n+1\} \\ j \neq i}} x_{ij}^k \leq b_{ik}, \forall i \in T, k \in O \tag{4}$$

$$x_{ij}^k(ws_{jk} - we_{ik} - st_{ij}^k) \geq 0, \forall i, j \in T, k \in O \tag{5}$$

$$\sum_{i \in T} \sum_{\substack{j \in T \cup \{n+1\} \\ j \neq i}} x_{ij}^k(we_{ik} - ws_{ik})m_k \leq M_k, \forall k \in O \tag{6}$$

$$\sum_{i \in T} \sum_{\substack{j \in T \cup \{n+1\} \\ j \neq i}} x_{ij}^k(we_{ik} - ws_{ik})e_k + \sum_{i \in T} \sum_{\substack{j \in T \\ j \neq i}} x_{ij}^k \cdot \rho_{ij}^k \leq E_k, \forall k \in O \tag{7}$$

$$x_{ij}^k \in \{0, 1\}, \forall i, j \in T \cup \{0, n+1\}, k \in O \tag{8}$$

The objective function (1) is to maximize the profits of the scheduled tasks. The set of constraints (2) guarantee that each task will be observed at most once. Then, constraints (3) ensure that the number of predecessors is equal to the number of successors for each task. Constraints (4) enforce that each task can only be scheduled to the orbits that are available for it. There must be sufficient setup time between consecutive tasks for transformations, which is enforced in constraints (5). Constraints (6) check that the memory consumption of the scheduled tasks cannot exceed the memory capacity for each orbit. Constraints (7) compute the energy consumption of the task sequence for each orbit, and enforce that the energy consumption must be less than or equal to the capacity.

3.2. A novel assignment formulation for deterministic EOS scheduling

In this section, we propose a novel assignment formulation for EOS scheduling, which applies the idea of a forbidden sequence to formulate the setup time and energy constraints linearly. In this formulation, we firstly sequence the available tasks for each orbit in time order according to the time windows. Hence, we can get an initial sequence of tasks for each orbit, and the EOS scheduling problem can be formulated as searching for a subsequence of tasks that belong to the initial sequence for each orbit, which will satisfy the constraints and maximize the profits. Decision variable $x_{ik} = 1$ represents that task i is allocated to the orbit k, and otherwise $x_{ik} = 0$. The integer linear programming model of this assignment formulation is given by:

$$max \sum_{i \in T} \sum_{k \in O} \omega_i \cdot x_{ik} \tag{9}$$

subject to

$$\sum_{k \in O} x_{ik} \leq 1, \forall i \in T \tag{10}$$

$$x_{ik} \leq b_{ik}, \forall i \in T, k \in O \tag{11}$$

$$x_{ik} + x_{jk} \leq 1, \forall i, j \in T, k \in O, \text{if task } i \text{ is the predecessor of task } j \text{ on orbit } k \text{ and } we_{ik} + st_{ij}^k > ws_{jk} \tag{12}$$

$$\sum_{i \in T} x_{ik}(we_{ik} - ws_{ik})m_k \leq M_k, \forall k \in O \tag{13}$$

$$\sum_{i \in Seq_{k,t}} x_{ik} \leq |Seq_{k,t}| - 1, \forall k \in O, Seq_{k,t} \in S_k \tag{14}$$

$$x_{ik} \in \{0, 1\}, \forall i \in T, k \in O \tag{15}$$

Similarly with the previous flow formulation, the objective (9) is to maximize the profits of the scheduled tasks. Constraints (10)-(11) guarantee that each task will be observed at most once, and must be scheduled on the orbits that are available for it, which is similar with constraints (2)-(4). Constraints (12) represent the set of setup time constraints, in which $we_{ik} + st_{ij}^k > ws_{jk}$ denotes that the setup time between task i and task j is not sufficient for the transformation. Therefore, we set $x_{ik} + x_{jk} \leq 1$, which represents that at most one of task i and task j is able to be scheduled on orbit k. Memory constraints (13) ensure that the memory consumption of the scheduled tasks must be less than or equal to the capacity for each orbit, which is analogous with constraints (6). Constraints (14) are the energy constraints, in which $Seq_{k,t}$ represents a forbidden sequence of orbit k, which violates the energy constraint. $|Seq_{k,t}|$ is the number of tasks in the forbidden sequence $Seq_{k,t}$, and S_k is the set of all forbidden sequences on orbit k. Hence, we set $\sum_{i \in Seq_{k,t}} x_{ik} \leq |Seq_{k,t}| - 1$, which ensures that not all the tasks of the forbidden sequence $Seq_{k,t}$ can be scheduled on orbit k due to exceeding the energy capacity.

Apparently, in contrast with the previous flow formulation, the assignment formulation has much fewer variables and much fewer constraints without considering the energy constraints. The model is inefficient for solving, because it will require exponential time to enumerate all the forbidden sequences for each orbit. Thus, lazy constraint generation will be taken into account in this paper. We firstly relax the energy constraints (14) and solve the relaxation problem, adding some of the energy constraints to the relaxation problem only when they are violated. The algorithm will be described in detail in Section 4.2.

3.3. A chance constraint programming model for EOS scheduling

Considering the uncertainties of clouds, the success of observations was formulated as stochastic events, and then we extend the assignment formulation to a chance constraint programming model.

Let $(1 - \alpha)$ denote the predefined confidence level. The objective (9) is then replaced by:

$$max \bar{f}, \tag{16}$$

in which $\bar{f}$ is constrained by the following chance constraint:

$$P\{\sum_{i \in T} \sum_{k \in O} \omega_i \cdot \tilde{\lambda}_{ik} \cdot x_{ik} \geq \bar{f}\} \geq 1 - \alpha \tag{17}$$

The chance constraint (17) states that the probability that the profits of observations under uncertainties of clouds will be at least $\bar{f}$ is larger than or

equal to the confidence level $1-\alpha$. Therefore, the CCP model is formulated as: $max\bar{f}$ subject to: (10)-(15),(17). This formulation is a modification of the previous assignment formulation, with the solution complexity amplified due to the addition of randomness.

4. Solution method

In this paper, we are the first to employ the CCP model to formulate the EOS scheduling problem under uncertainties of clouds. We know that this model is intractable to solve due to the following difficulties:

- The probability in the chance constraint (17) is very hard to calculate, and hence just checking the feasibility of a solution is already difficult [37, 38].
- The feasible region defined by the chance constraint is generally not convex [37, 38].
- The number of forbidden sequences is exponential in the number of tasks.

The first two difficulties will be tackled by sample approximation, and the last one will be approached by a branch and cut algorithm.

4.1. Sample approximation

The main idea of sample approximation is to solve the chance constraint programming problem as follows. Firstly, we create a sample $w_1, ..., w_N$ of scenarios (realizations) of the random vector $w(\tilde{\lambda}_{ik}), i \in T, k \in O$ by Monte Carlo simulation. In this way, the original distribution of the random vector $w(\tilde{\lambda}_{ik})$ is replaced with the empirical distribution of the sample. Under some conditions, a feasible solution of the sample approximation will also be feasible for the original CCP problem with a high probability.

4.1.1. Sample approximation formulation

A sample W is a set of scenarios of the random vector $w(\tilde{\lambda}_{ik}), i \in T, k \in O$, such that $W = \{w_1, ..., w_{|W|}\}$, in which $|W|$ is the sample size. The basic idea of the reformulation introduced in [42] is to solve the problem and get a solution, which is infeasible for at most $\lfloor |W| \cdot \epsilon \rfloor$ scenarios. Thus the solution will be feasible for the sample approximation problem with a confidence level at least $(1-\epsilon)$. On the basis of that idea, we reformulate the CCP model as below:

Let $y_l, w_l \in W$ be binary variables, $y_l = 0$ if the obtained solution must be feasible for scenario w_l and $y_l = 1$ otherwise.

Chance constraint (17) can be replaced by the following constraints:

$$\sum_{i \in T} \sum_{k \in O} \omega_i \cdot \lambda_{ik}^l \cdot x_{ik} \geq -y_l M + \bar{f}, \forall w_l \in W \tag{18}$$

$$\sum_{w_l \in W} y_l \leq \lfloor |W| \cdot \epsilon \rfloor \tag{19}$$

$$y_l \in \{0,1\}, \forall w_l \in W \tag{20}$$

Constraints (18) state that the profits of the observations have to be larger than or equal to $\bar{f}$ for scenario w_l if $y_l = 0$, in which λ_{ik}^l is the value of stochastic variable $\tilde{\lambda}_{ik}$ under scenario w_l, and M is assumed to be a sufficiently large number, which can be set to the sum of the profits of all the tasks. Constraint (19) imposes that the number of scenarios for which the solution is not necessarily feasible, which means the profits of the observations are not necessary to be larger than or equal to $\bar{f}$, will be at most $\lfloor |W| \cdot \epsilon \rfloor$. Thus, the sample approximation formulation of the original CCP problem is: maximize $\bar{f}$ subject to (10)-(15), (18)-(20).

4.1.2. Sample size

A larger sample size implies a better approximation. However, a larger sample size also implies a more difficult and inefficient to solve sample approximation problem. Hence, we should make a good trade-off between the approximation quality and the solving efficiency. In previous research, [37] computed a lower bound of the sample size which guarantees that the feasible solution of the sample approximation problem will also be feasible for the original CCP problem with a certain probability.

Suppose that $1 - \epsilon > 1 - \alpha$ and $(1 - \theta)$ be the probability that a feasible solution of the sample approximation problem will yield a feasible solution of the original CCP problem. Let U be determined by $|X| \leq U^n$ and n is the number of decision variables. Besides, because the decision variables $x_{ik}, i \in T, k \in O$ are binary variables, U will be set to 2. Afterwards, the sample size $|W|$ can be determined by the following expression [37]:

$$|W| \geq \frac{1}{2(\alpha - \epsilon)^2} log(\frac{1}{\theta}) + \frac{n}{2(\alpha - \epsilon)^2} log(U) \tag{21}$$

It has been described in [37] that the sample size from the above expression will be too conservative. Most of the times, much smaller samples will be sufficient to guarantee the feasibility.

Hence, in this study, we firstly solve the sample approximation problem with a smaller sample W_1 (e.g. $|W_1| = 100$), getting a solution S. Afterwards, we will test the obtained solution S on a larger sample W_2 (e.g. $|W_2| = 1000$) to estimate the real confidence level reached. The estimated confidence level reached can be calculated by the number of scenarios belonging to sample W_2 for which S is feasible, divided by the sample size $|W_2|$.

4.2. Branch and cut

A branch and cut algorithm is a generalization of a branch and bound (B&B) algorithm, which introduces the valid inequalities as cutting planes (cuts) in the nodes of the B&B tree. After solving the LP relaxation problem, if not successful in pruning the node, B&C will try to find violated cuts. Consequently, if one or more violated cuts are found, they will be added to the LP relaxation formulation and be solved again, otherwise B&C will branch. Typically, the

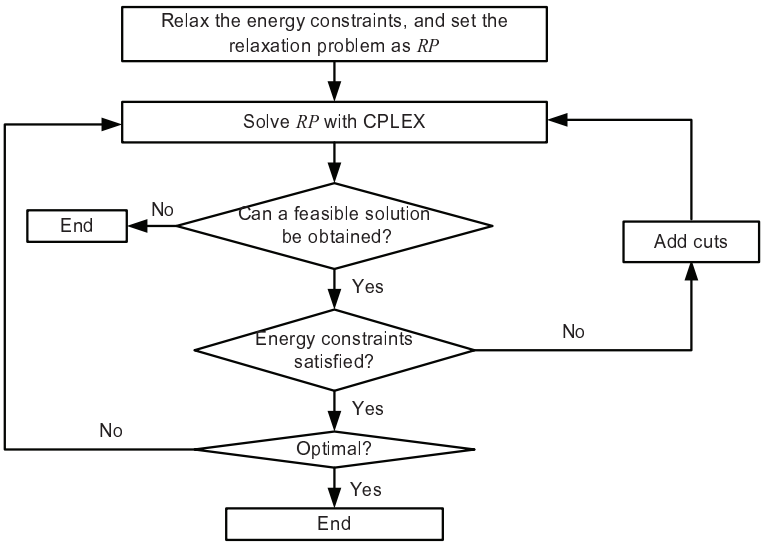

Figure 3: Branch and cut algorithm: OP1

cuts introduced will make the LP relaxation based bound tighter, which implies a smaller B&B tree and faster convergence to the optimal solution. For the details and applications of a B&C algorithm, readers are referred to [2, 11, 40].

The idea of lazy constraint generation is to add the constraints that define the feasible solutions to the formulation when they are violated. Since the number of forbidden sequences is exponential in the number of tasks, it will be impractical to introduce the complete set of energy constraints (14). Hence, on the basis of lazy constraint generation, we design a B&C algorithm, which leaves out the forbidden sequence constraints, and solves the relaxation problem by CPLEX. Subsequently, the forbidden sequence constraints will be added as cuts when they are violated. With different ways of adding lazy constraints, there will be two options for the B&C algorithm:

- First, the B&C algorithm will add the required lazy constraints when it finds an arbitrary integer feasible solution. In the search of B&B tree, if we get a feasible solution, say S, we will check whether all the energy constraints have been satisfied. If some constraints are violated, we will add the respective forbidden sequence constraints, which are shown in (14), as cuts to the relaxation problem, and continue to solve the new problem. Otherwise, if all constraints are satisfied and S is the optimal solution, which means that S is optimal for the original SA problem, the B&C algorithm will end successfully; if S is not optimal, we will go on with the search to find the optimal solution. We will refer to this option as *OP1*, which is shown in Figure 3.

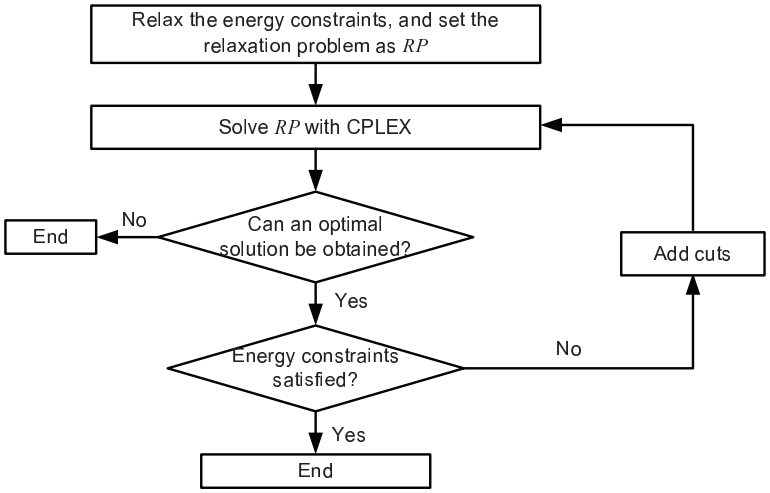

Figure 4: Branch and cut algorithm: OP2

- Different from *OP1*, a second option is to add the lazy constraints only when the B&C algorithm obtains the optimal solution. If all the constraints are satisfied, the solution will be optimal for the original SA problem, and the B&C algorithm will end successfully; otherwise, we will add the respective forbidden sequence constraints as cuts, and solve the new problem optimally again. We call this option *OP2*, which is illustrated in Figure 4.

Algorithm 1 Lazy constraint generation

1: **for** each orbit k **do**
2: Assume the scheduled task sequence of the current solution on orbit k is $i(1), i(2), ..., i(n)$
3: **if** $\sum_{j=1}^{n-1}[(we_{i(j),k} - ws_{i(j),k})e_k + \rho^k_{i(j),i(j+1)}] + (we_{i(n),k} - ws_{i(n),k})e_k > E_k$ **then**
4: Add cut (constraint): $\sum_{j=1}^{n} x_{i(j),k} \leq n-1$.
5: **end if**
6: **end for**

Algorithm 1 simply describes how to generate the forbidden sequence constraints and to add the constraints as cuts. If a feasible solution of the relaxation problem is obtained from CPLEX, we will have a task sequence for each orbit. The orbits on which no tasks are scheduled will have empty sequences. For each orbit, if the energy consumption of the sequence of tasks is larger than the energy capacity, this task sequence represents a forbidden sequence, and we will set the energy constraint $\sum_{j=1}^{n} x_{i(j),k} \leq n-1$, which denotes that not all the tasks in the forbidden sequence can be scheduled on this orbit.

Table 2: A toy EOS scheduling instance

Task no.	Profit	Orbit no.		
		1	2	3
1	7	[42,48]	-	[66,72]
2	9	-	[16,20]	-
3	3	[18,25]	[42,49]	-
4	6	[50,55]	[66,71]	-
5	6	-	-	[32,38]
6	7	[22,27]	[41,46]	[57,62]
7	9	-	[27,34]	[42,47]
8	8	[32,38]	[53,59]	[18,24]
Memory capacity		60	40	50
Energy capacity		80	50	70
Memory consumption for each unit time		2.5	2	2.5
Energy consumption for each unit time		1.5	2	2

Example

To describe the proposed assignment formulation and the B&C algorithm more clearly, let us introduce a toy instance of EOS scheduling. In order to facilitate the description, we will not consider the uncertainties of clouds, and thus the tasks will be successfully observed if they are scheduled. This assumption does not affect how the B&C algorithm essentially works. The instance is composed of 8 non-dummy tasks and 3 orbits, and Table 2 outlines the following settings: profits of tasks, availabilities for observations, time windows, memory and energy capacities, as well as memory and energy consumptions for each unit observation time. The symbol "-" denotes that the orbit is not available for observing the task.

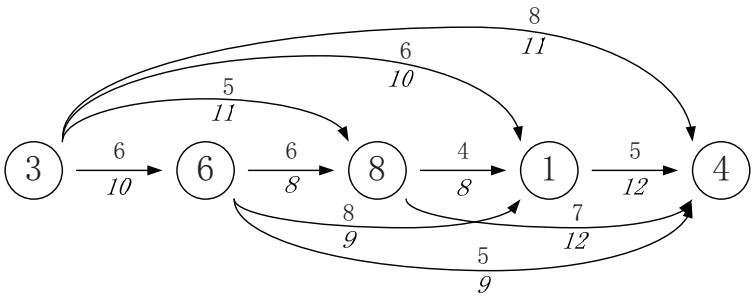

Figure 5: Setup times and slewing energy for orbit 1

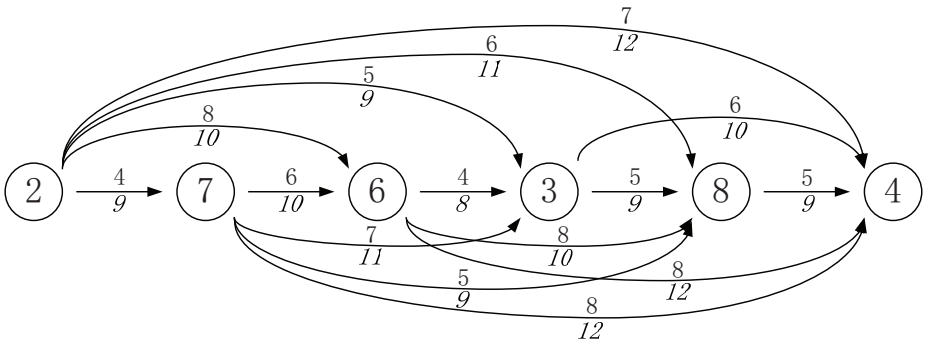

Figure 6: Setup times and slewing energy for orbit 2

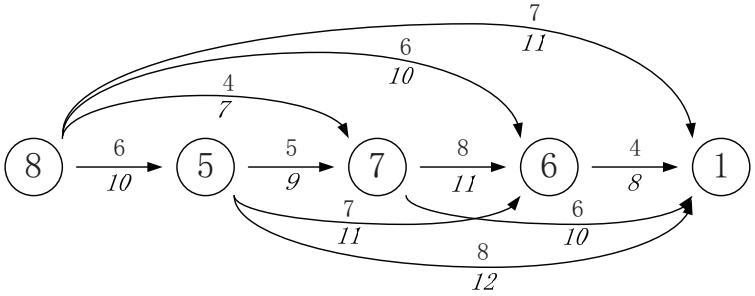

Figure 7: Setup times and slewing energy for orbit 3

According to the time order of the time windows, we can sequence the available tasks for each orbit, which are shown as follows:

$$\begin{aligned}
&\text{Orbit } 1: \ 3 \to 6 \to 8 \to 1 \to 4\\
&\text{Orbit } 2: \ 2 \to 7 \to 6 \to 3 \to 8 \to 4\\
&\text{Orbit } 3: \ 8 \to 5 \to 7 \to 6 \to 1
\end{aligned}$$

On the basis of the above sequences, the EOS scheduling problem will be formulated as searching for a subsequence of tasks for each orbit, satisfying the constraints and maximizing the profits. In addition, based on the above sequences, the setup times and slewing energy between tasks are illustrated in Figures 5-7, in which the normal numbers above the lines denote the setup times and the italic numbers below the lines denote the slewing energy.

Based on the sequences, we can easily formulate the setup time constraints. For example, let us consider tasks 3 and 6 on orbit 1: we get the inequality that $we_{3,1} + st^1_{3,6} > ws_{6,1}$, which represents that the setup time is not sufficient for the transformation from task 3 to task 6 on orbit 1. Hence, we will set the setup time constraint as follows:

$$x_{3,1} + x_{6,1} \leq 1,$$

which forbids that both task 3 and task 6 are scheduled on orbit 1. In the following, we show how the B&C algorithm with lazy constraint generation works. Firstly, the B&C algorithm solves the relaxation problem with CPLEX, leaving out the energy constraints. And then, we will get a feasible solution of the relaxation problem, which is the following:

$$\begin{aligned}&\text{Orbit 1}: \ 3 \rightarrow 4\\ &\text{Orbit 2}: \ 2 \rightarrow 7 \rightarrow 8\\ &\text{Orbit 3}: \ 5 \rightarrow 6 \rightarrow 1.\end{aligned}$$

For orbit 2, with **Algorithm 1**, we can compute the energy consumption as 52, which is larger than the energy capacity of orbit 2. Hence, the task sequence $2 \rightarrow 7 \rightarrow 8$ is a forbidden sequence on orbit 2, and thus we will formulate the forbidden sequence constraint in the following way:

$$x_{2,2} + x_{7,2} + x_{8,2} \leq 2,$$

which indicates that not all the tasks in the forbidden sequence can be scheduled on this orbit. After adding the above constraint to the relaxation problem as a cut, we will return to CPLEX to solve the new problem, and then we will get a new feasible solution:

$$\begin{aligned}&\text{Orbit 1}: \ 3 \rightarrow 8 \rightarrow 4\\ &\text{Orbit 2}: \ 2 \rightarrow 7\\ &\text{Orbit 3}: \ 5 \rightarrow 6 \rightarrow 1\end{aligned}$$

where task 8 has been rescheduled on orbit 1. From CPLEX, we obtain that the feasible solution above has been the optimal solution of the relaxation problem and we can calculate that the energy constraints have been satisfied for each orbit. Thus this solution is the optimal solution of the original sample approximation problem, and the B&C algorithm ends.

5. Computational results

For this section, we created a great number of problem instances in order to evaluate the effectiveness and efficiency of our proposed approach. The computational tests have three goals: compare two different options of the B&C algorithm, verify the effectiveness of sample approximation, and evaluate the efficiency of the assignment formulation and the B&C algorithm.

To verify the effectiveness and efficiency of our algorithm, the tasks are randomly generated in the area: latitude 0°-60° and longitude 0°-150°. Without loss of generality, the priorities of tasks are integers, uniformly distributed in the interval [1,10]. In correspondence with the literature [4, 10, 23, 41], three different satellites are considered in this paper. The parameters of the satellites are outlined in Table 3, and the orbit models of the satellites are obtained from the Satellite Tool Kit (STK). In addition, the memory capacity and energy capacity for each orbit are randomly generated in the intervals [100,120] and [120,160], respectively. Considering the uncertainties of clouds, for each time window of

Table 3: Parameters of satellites

Satellite	Slewing velocity	Startup time	Shutdown time	Stability time	Memory /time	Energy /time	Energy /deg
CBERS-2	2	5	8	3	2	1.5	1.5
IKONOS-2	2.5	8	5	6	4	2.5	4
SPOT-5	3	10	10	9	3	3.5	1

observation, the probability that there is no cloud block, i.e. the observation is successful, will be set in the interval [0.5,1].

The B&C algorithm was implemented in C++ using the CPLEX 12.3 API and ran on a server equipped with an Intel(R) Xeon(R) CPU E5-2690 0@ 2.90 GHz (2 processors) and 128 Gb RAM, with operating system Window Server 2012 Standard.

5.1. Comparison between OP1 and OP2

Our first computational experiment was ran for comparing the performance of the options OP1 and OP2. In this experiment, the numbers of tasks are 20, 40 and 60, respectively. The scheduling horizons are set to be 12 and 24 hours, which are corresponding to 21 and 42 orbits, respectively. For each combination of task number and orbit number, we create 10 problems randomly. Besides, the sample size is set to be 50 or 100. Table 4 shows the comparison results obtained. Column "Obj" contains the average of the scheduling objective values of $\bar{f}$ for the 10 instances, and "Time" shows the average values of the solution times.

According to the results in Table 4, we can conclude that OP1 performs better than OP2. In this experiment, all the problems are solved optimally. Hence, the scheduling objective values are equivalent all the time, which are the optimal solutions. However, OP1 is much faster to obtain the optimal solutions, which is more efficient. Consequently, in the remainder of this paper, we will only adopt OP1 for the B&C algorithm.

5.2. Effectiveness verification of the sample approximation

To verify the effectiveness of the sample approximation method, in this section, we set the number of tasks to be 20, 40, 60 and 80, respectively, and the number of orbits is 21. For each number of tasks, we generate 5 problem instances randomly. Thus we have 20 problem instances in all. In addition, the confidence level of the original chance constraint programming problem is 0.9, and the confidence level of the sample approximation is 0.99 and 0.95, alternatively. We set the sample size for solving the chance constraint programming problem to be 100 and 200, respectively. For each problem and for each combination of sample

Table 4: Comparison results for OP1 and OP2

Number of orbits	Number of tasks	Sample size	OP1		OP2	
			Obj	Time	Obj	Time
21	20	50	52.8	0.322	52.8	0.628
		100	53.2	0.564	53.2	1.053
	40	50	109.0	0.438	109.0	3.647
		100	104.7	0.832	104.7	6.915
	60	50	160.2	0.599	160.2	10.076
		100	159.7	1.463	159.7	19.658
42	20	50	73.9	0.361	73.9	0.874
		100	72.4	1.294	72.4	3.538
	40	50	154.5	0.794	154.5	3.816
		100	151.8	3.834	151.8	20.838
	60	50	240.7	2.732	240.7	35.426
		100	236.1	27.604	236.1	355.484

Table 5: Parameters for simulation study I

Parameter	Values
Number of tasks n	20, 40, 60, 80
Number of orbits m	21
Number of problem instances	5
Confidence level of CCP $1-\alpha$	0.9
Sample confidence level $1-\epsilon$	0.99, 0.95
Sample size $\|W\|$	100, 200
Number of runs	100

confidence level and sample size, we solve the problem 100 times, which are called 100 runs. For each run, we test the solution of the sample approximation approach on a larger sample with the sample size being 1000. Then, we will compare the following statistics: the minimal, average and maximum really reached confidence level of the 100 runs, respectively. We also count the number of feasible runs, with the really reached confidence level larger than or equal to the confidence level of the original CCP problem. The simulation parameters and their values are outlined in Table 5.

It is indicated in Table 6 that when the number of tasks is 20 and the sample confidence level $1-\epsilon$ is 0.99, a sample size of 100 is sufficient because most runs are feasible. Besides, if the sample size is 200, all the runs will be feasible. However, if $1-\epsilon$ is set to be 0.95, a sample size of 100 will not be enough, because many runs are infeasible. Thus the sample size should be increased to

Table 6: Effectiveness and feasibility of sample approximation method

Number of tasks	Instance no.	$1-\epsilon$	Sample size 100				Sample size 200			
			Min	Ave	Max	Num	Min	Ave	Max	Num
20	1	0.99	0.907	0.948	0.986	100	0.926	0.964	0.990	100
		0.95	0.830	0.903	0.965	56	0.873	0.928	0.963	89
	2	0.99	0.910	0.961	0.989	100	0.954	0.972	0.988	100
		0.95	0.833	0.910	0.971	68	0.868	0.930	0.973	88
	3	0.99	0.815	0.946	0.987	90	0.920	0.962	0.986	100
		0.95	0.825	0.898	0.948	53	0.869	0.922	0.960	88
	4	0.99	0.902	0.944	0.979	100	0.906	0.960	0.989	100
		0.95	0.806	0.892	0.960	45	0.871	0.924	0.970	87
	5	0.99	0.899	0.940	0.979	99	0.915	0.961	0.988	100
		0.95	0.802	0.884	0.941	38	0.847	0.915	0.955	78
40	1	0.99	0.907	0.948	0.986	100	0.926	0.964	0.990	100
		0.95	0.763	0.893	0.963	46	0.864	0.913	0.957	78
	2	0.99	0.864	0.933	0.971	87	0.932	0.963	0.987	100
		0.95	0.786	0.895	0.956	47	0.884	0.919	0.964	81
	3	0.99	0.873	0.926	0.960	82	0.906	0.952	0.979	100
		0.95	0.757	0.870	0.959	28	0.858	0.905	0.935	63
	4	0.99	0.880	0.953	0.998	94	0.923	0.973	0.995	100
		0.95	0.836	0.924	0.972	84	0.898	0.936	0.968	96
	5	0.99	0.881	0.928	0.976	91	0.902	0.953	0.977	100
		0.95	0.796	0.871	0.948	19	0.824	0.900	0.949	74
60	1	0.99	0.860	0.923	0.968	80	0.914	0.949	0.975	100
		0.95	0.787	0.870	0.939	17	0.852	0.906	0.939	67
	2	0.99	0.857	0.931	0.971	89	0.917	0.955	0.979	100
		0.95	0.763	0.875	0.948	29	0.872	0.913	0.943	78
	3	0.99	0.829	0.921	0.974	82	0.921	0.956	0.983	100
		0.95	0.821	0.878	0.930	23	0.850	0.907	0.953	66
	4	0.99	0.877	0.932	0.979	92	0.912	0.961	0.984	100
		0.95	0.797	0.878	0.944	28	0.875	0.920	0.972	84
	5	0.99	0.853	0.916	0.964	75	0.908	0.950	0.974	100
		0.95	0.749	0.854	0.929	12	0.865	0.902	0.930	61
80	1	0.99	0.788	0.920	0.982	75	0.902	0.948	0.977	100
		0.95	0.745	0.864	0.944	21	0.878	0.906	0.939	59
	2	0.99	0.840	0.907	0.955	62	0.894	0.944	0.980	98
		0.95	0.807	0.868	0.926	11	0.842	0.899	0.951	46
	3	0.99	0.845	0.906	0.966	54	0.927	0.951	0.977	100
		0.95	0.823	0.877	0.94	19	0.876	0.913	0.936	77
	4	0.99	0.822	0.899	0.967	49	0.908	0.946	0.975	100
		0.95	0.792	0.849	0.904	5	0.865	0.895	0.926	39
	5	0.99	0.862	0.911	0.962	68	0.909	0.946	0.976	100
		0.95	0.753	0.859	0.924	14	0.857	0.903	0.956	62

200. Then, if there are 40 tasks, for $1-\epsilon=0.99$, the sample size being 100 is sufficient, but being 200 is better because all runs are feasible. Besides, if $1-\epsilon=0.95$, the sample size needs to be 200 at least. When the number of tasks is 60, for $1-\epsilon=0.99$, the sample size can be either 100 or 200, and if the confidence level is 0.95, both 100 and 200 are not feasible, which implies that we need a larger sample. Similarly with the previous one, for 80 tasks, if $1-\epsilon$ is 0.95, neither 100 nor 200 is sufficient as a sample size, so we need a larger sample. From Table 6, it can be concluded that if we set the sample confidence level larger (e.g. $1-\epsilon=0.99$), we can get feasible solutions using a smaller sample. In contrast, if the sample confidence level is smaller (e.g. $1-\epsilon=0.95$), we will need a larger sample to guarantee the feasibility.

5.3. Effectiveness verification of the branch and cut algorithm

In this section, we conduct some simulation experiments to verify the effectiveness and efficiency of our proposed assignment formulation and B&C algorithm. Firstly, for a fair comparison, we extend the flow formulation, described in section 3.1, to a chance constraint programming model under uncertainties of clouds. Similarly with the assignment formulation, the objective of flow formulation (1) is replaced by:

$$max\bar{f} \tag{22}$$

where $\bar{f}$ is constrained by the subsequent chance constraint:

$$P\{\sum_{i\in T}\sum_{\substack{j\in T\cup\{n+1\}\\ j\neq i}}\sum_{k\in O}\omega_i\cdot\tilde{\lambda}_{ik}\cdot x_{ij}^k\geq\bar{f}\}\geq 1-\alpha \tag{23}$$

The chance constraint (23) indicates that the probability that the profits of observations under uncertainties of clouds will be at least $\bar{f}$ is larger than or equal to the confidence level $1-\alpha$. Therefore, the CCP model of the flow formulation is depicted as: $max\bar{f}$ subject to (2)-(8), (23). To solve the chance constraint problem, we need to transform it to the sample approximation problem. Let W be a sample of scenarios of the random vector $w\{\tilde{\lambda}_{ik}\}, i\in T, k\in O$, such that $W=\{w_1,...,w_{|W|}\}$, where $|W|$ is the sample size. $y_l, w_l\in W$ are binary variables, $y_l=0$ if the obtained solution must be feasible for scenario w_l and $y_l=1$ otherwise. Analogously to the assignment formulation, the chance constraint (23) can be replaced by the following constraints:

$$\sum_{i\in T}\sum_{\substack{j\in T\cup\{n+1\}\\ j\neq i}}\sum_{k\in O}\omega_i\cdot\lambda_{ik}^l\cdot x_{ij}^k\geq -y_lM+\bar{f},\forall w_l\in W \tag{24}$$

$$\sum_{w_l\in W}y_l\leq\lfloor|W|\cdot\epsilon\rfloor \tag{25}$$

$$y_l\in\{0,1\},\forall w_l\in W \tag{26}$$

Hence, the sample approximation formulation of the CCP problem of the flow formulation is: maximize $\bar{f}$, subject to (2)-(8), (24)-(26). Clearly, the

Table 7: Parameters for simulation study II

Parameter	Values
Number of tasks n	20, 40, 60, 80, 100, 120
Number of orbits m	21, 42
Number of problem instances	10
Confidence level of CCP $1-\alpha$	0.9
Sample confidence level $1-\epsilon$	0.99
Sample size $\|W\|$	50, 100, 200

above sample approximation problem is an integer linear programming problem, which can be immediately solved by CPLEX. In the following, the results of this formulation solved with CPLEX will be compared with those of our proposed assignment formulation and B&C algorithm.

In this experiment, the number of tasks ranges from 20 to 120 with an increment of 20, and the number of orbits are 21 and 42, respectively. Besides, for each combination of task number and orbit number, we randomly create 10 problem instances. Therefore, we have 120 instances in all. The sample size to solve the problems is 50, 100 and 200, respectively. In addition, the time limit for solving each instance is fixed at 900 seconds. Afterwards, we will compare the following statistics: the objective value $\bar{f}$ and the solution time. In addition, for each combination of task number and orbit number, we will also count the number of instances that are solved optimally, for which we can only get feasible solutions, and which we cannot solve due to the out-of-memory status of the computer, respectively. The simulation parameters and their values are outlined in Table 7.

Table 8 shows the comparison results when the number of orbits equals 21. Similarly with Table 4, column "Obj" contains the average of the schedule objective values of $\bar{f}$ for the 10 instances, and "Time" contains the average values of the solution times. Besides, "(opt,fea)" shows the number of instances that are solved optimally and the number of instances for which we can only get feasible solutions, respectively. Furthermore, for the other problem instances that are not included in (opt,fea), we cannot get solutions. It is illustrated that when the number of orbits is 21, almost all the problem instances are solved optimally with only a few exceptions, which are denoted in italic numbers. In addition, for the majority of the instances, the assignment formulation and B&C algorithm are faster to solve the problem optimally. However, there are some exceptions where the B&C algorithm is slower than solving the problem directly by CPLEX. These are denoted with underlined numbers. The reason for this is that with the increase of the number of tasks the energy constraints based on the forbidden sequences will increase drastically. Hence, there will be more loops to add cuts, which will make the B&C algorithm slower and less efficient. However, it doesn't imply that the B&C algorithm performs worse when the

Table 8: Scheduling objective, solving time and the number of instances of (optimal, feasible) solutions for 21 orbits

Number of tasks	Sample size	Assignment (B&C)			Flow (CPLEX)		
		Obj	Time	(opt,fea)	Obj	Time	(opt,fea)
20	50	53.4	0.610	(10,0)	53.4	0.940	(10,0)
	100	51.2	0.790	(10,0)	51.2	1.110	(10,0)
	200	50.7	1.681	(10,0)	50.7	4.050	(10,0)
40	50	107.0	1.571	(10,0)	107.0	3.232	(10,0)
	100	104.9	1.839	(10,0)	104.9	4.373	(10,0)
	200	102.9	4.817	(10,0)	102.9	20.438	(10,0)
60	50	162.3	2.882	(10,0)	162.3	3.599	(10,0)
	100	159.6	3.181	(10,0)	159.6	12.003	(10,0)
	200	155.2	13.633	(10,0)	155.2	41.445	(10,0)
80	50	193.0	4.700	(10,0)	193.0	13.200	(10,0)
	100	188.0	9.120	(10,0)	188.0	32.800	(10,0)
	200	185.4	151.039	(10,0)	185.4	178.726	(10,0)
100	50	231.1	9.349	(10,0)	231.1	17.875	(10,0)
	100	225.4	20.296	(10,0)	225.4	61.609	(10,0)
	200	222.1	317.959	(10,0)	222.1	271.865	(10,0)
120	50	261.1	28.028	(10,0)	261.1	39.900	(10,0)
	100	*255.4*	*264.017*	*(8,2)*	255.4	231.152	(10,0)
	200	*250.4*	*743.799*	*(3,7)*	*250.5*	*587.845*	*(7,3)*

Table 9: Scheduling objective, solving time and the number of instances of (optimal, feasible) solutions for 42 orbits

Number of tasks	Sample size	Assignment (B&C)			Flow (CPLEX)		
		Obj	Time	(opt,fea)	Obj	Time	(opt,fea)
20	50	73.8	0.770	(10,0)	73.8	1.240	(10,0)
	100	71.5	1.210	(10,0)	71.5	3.210	(10,0)
	200	69.7	2.077	(10,0)	69.7	5.273	(10,0)
40	50	155.4	1.763	(10,0)	155.4	4.934	(10,0)
	100	152.7	3.991	(10,0)	152.7	21.057	(10,0)
	200	148.6	11.408	(10,0)	148.6	31.896	(10,0)
60	50	239.4	6.070	(10,0)	239.4	18.243	(10,0)
	100	234.6	21.849	(10,0)	234.6	85.196	(10,0)
	200	*231.8*$^+$	*259.106*	*(9,1)**	*231.7*$^+$	*324.657*	*(8,2)**
80	50	320.0	14.800	(10,0)	320.0	59.200	(10,0)
	100	*317.0*	*175.000*	*(9,1)**	*317.0*	*377.000*	*(8,2)**
	200	*309.1*	*534.972*	*(6,4)**	*309.1*	*731.798*	*(5,5)**
100	50	*378.2*$^+$	*177.385*	*(9,1)*	*378.1*$^+$	*278.822*	*(9,1)*
	100	*374.6*$^+$	*669.627*	*(4,6)**	*374.1*$^+$	*832.304*	*(2,8)**
	200	**366.5**$^+$	**892.164**	**(1,7)***	**362.0**$^+$	**900**	**(0,4)***
120	50	**447.0**	**<u>332.232</u>**	**(7,2)**	**447.0**	**<u>285.534</u>**	**(9,0)**
	100	*448.8*$^+$	*826.166*	*(2,8)*	*447.8*$^+$	*849.453*	*(2,8)*
	200	**440.1**$^+$	**900**	**(0,10)***	**408.0**$^+$	**900**	**(0,2)***

problem size is large, because the B&C algorithm will save more memory for solving. Therefore, the B&C algorithm will obtain more optimal or feasible solutions with respect to large-scale problems, as shown in Table 9. With an increase of the sample size, the objective value $\bar{f}$ decreases, because the value of $\bar{f}$ should be smaller to guarantee that more scenarios are feasible. Besides, with the increase of the sample size, both formulations will have more variables and constraints. Thus the solution times increase, and fewer instances can be solved optimally in the limited time.

The experimental results when the number of orbits is 42 are outlined in Table 9. The notations of each column are the same as those in Table 8. It reveals that for many problem instances we cannot find optimal solutions, that for the remaining problems we can sometimes find feasible solutions (denoted by italic numbers), while for some problems we cannot even find a feasible solution (denoted by boldfaced numbers). It must be noted that the average values of the objective values and the solution times are only computed on the solvable instances. As shown in the table, if the instances are solved optimally, the branch and cut algorithm will be faster to obtain the optimal solutions. Similarly with Table 8, for some instances that cannot be solved optimally, there exist some exceptions where the B&C algorithm is slower than CPLEX (denoted

by underlined numbers). However, for most instances, the B&C algorithm can obtain better solutions in less time, which implies a larger objective value for the feasible solutions (denoted by +) and more optimal or feasible solutions among the 10 instances (denoted by *).

6. Conclusions and future work

In this paper, considering the uncertainties of clouds, we formulated the cloud blocks for observations as stochastic events, and then investigated the scheduling of multiple EOSs. After the comparisons of time-indexed, flow and assignment formulations, we proposed a novel assignment formulation with the inspiration of the notion of a forbidden sequence, which has less variables and less constraints. Subsequently, under the uncertainties of clouds, we extended the assignment formulation to a chance constraint programming model. To solve the CCP model, we transformed the model into an integer linear programming model by sample approximation. Afterwards, with lazy constraint generation, we suggested a branch and cut algorithm to solve the sample approximation problem. Finally, a great number of simulation experiments were conducted to verify the effectiveness and feasibility of the sample approximation method and the B&C algorithm.

In the future, we will consider the scheduling of agile EOSs under uncertainties. Different from the non-agile satellites in this study, the agile satellites do not only have the maneuverability of slewing, but also the maneuverability of pitching, along with the orbit. Hence, the satellite will have a long time window for observation. Consequently, we need not only allocate the tasks to the orbits, but also need to decide the start and finish times. In addition, for a unique window, the impact of clouds for different parts will be different, which will make the problem more complicated.

References

[1] Agnése JC, Bensana E. Exact and approximate methods for the daily management of an earth observation satellite. In: Proceeding of the fifth ESA workshop on aritificial intelligence and knowledge based systems for space; 1995.

[2] Bard JF, Kontoravdis G, Yu G. A branch-and-cut procedure for the vehicle routing problem with time windows. Transp Sci 2002; 36(2): 250-269.

[3] Benoist T, Rottembourg B. Upper bounds for revenue maximization in a satellite scheduling problem. 4OR-Q J Oper Res 2004; 2(3): 235-249.

[4] Bensana E, Verfaillie G, Agnese JC, Bataille N, Blumestein D. Exact and inexact methods for the daily management of an earth observation satellite. In: Proceeding of the international symposium on space mission operations and ground data systems; 1996. 4: 507-514.

[5] Bensana E, Verfaillie G, Michelon-Edery C, Bataille N. Dealing with uncertainty when managing an earth observation satellite. In: Proceeding of the fifth international symposium on artificial intelligence, robotics and automation in space (ISAIRAS'99); 1999, p.205-210.

[6] Bianchessi N, Piuri V, Righini G, Roveri M, Laneve G, Zigrino A. An optimization approach to the planning of earth observing satellites. In: Proceeding of the fourth international workshop on planning and scheduling for space (IWPSS'04); 2004.

[7] Bianchessi N, Righini G. A mathematical programming algorithm for planning and scheduling an earth observing SAR constellation. In: Proceeding of the fifth international workshop on planning and scheduling for space (IWPSS'06); 2006.

[8] Bianchessi N, Cordeau JF, Desrosiers J, Laporte G, Raymond V. A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites. Eur J Oper Res 2007; 177(2): 750-762.

[9] Bianchessi N, Righini G. Planning and scheduling algorithms for the COSMO-SkyMed constellation. Aerosp Sci Technol 2008; 12(7): 535-544.

[10] Cordeau J, Laporte G. Maximizing the value of an earth observation satellite orbit. J Oper Res Soc 2005; 56(8): 962-968.

[11] Fischetti M, Salazar Gonzalez JJ, Toth P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper Res 1997; 45(3): 378-394.

[12] Frank J, Jónsson A, Morris R, Smith DE. Planning and scheduling for fleets of earth observing satellites. In: Proceeding of the sixth international symposium on artificial intelligence, robotics, automation and space (ISAIRAS'01); 2001.

[13] Gabrel V, Moulet A, Murat C, Paschos VT. A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts. Ann Oper Res 1997; 69: 115-134.

[14] Gabrel V, Vanderpooten D. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. Eur J Oper Res 2002; 139(3): 533-542.

[15] Gabrel V, Murat C. Mathematical programming for earth observation satellite mission planning. In: Ciriani TA, Fasano G, Gliozzi S, Tadei R, editors. Operations research in space and air, Springer; 2003, p.103-122.

[16] Gabrel V. Strengthened 0-1 linear formulation for the daily satellite mission planning. J Comb Optim 2006; 11(3): 341-346.

[17] Globus A, Crawford J, Lohn J, Morris R. Scheduling earth observing fleets using evolitionary algorithms: Problem description and approach. In: Proceedding of the third international NASA workshop on planning and scheduling for space; 2002.

[18] Globus A, Crawford J, Lohn J, Pryor A. A comparison of techniques for scheduling fleets of earth-observing. J Oper Res Soc 2003; 56(8): 962-968.

[19] Habet D, Vasquez M. Solving the selecting and scheduling satellite photographs problem with a consistent neighborhood heuristic. In: Proceeding of the sixteenth IEEE international conference on tools with artificial intelligence (ICTAI'04); 2004.

[20] Habet D. Tabu search to solve real-life combinatorial optimization problems: A case of study. In: Abraham A, Hassanien AE, Carvalho AP, editors. Foundations of Computational Intelligence, Springer 2009; 3: 129-151.

[21] Habet D, Vasquez M, Vimont Y. Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite. Comput Optim Appl 2010; 47(2): 307-333.

[22] Hall NG, Magazine MJ. Maximizing the value of a space mission. Eur J Oper Res 1994; 78(2): 224-241.

[23] Han S, Beak S, Cho K, Lee D, Kim H. Satellite mission scheduling using genetic algorithm. In: Proceeding of the international conference on instrumentation, control and information technology (SICE'08); 2008.

[24] He M, He R. Research on agile imaging satellites schedllling techniques with the consideration of cloud cover. Sci Technol Eng 2013; 13(28): 8373-8379. (in chinese)

[25] Igelmund G, Radermacher FG. Preselective strategies for the optimization of stochastic scheduling problems. Networks 1983; 13: 1-28.

[26] Igelmund G, and Radermacher FG. Algorithmic approches to preselective strategies for stochastic scheduling problems. Networks 1983; 13: 29-48.

[27] Lemaître, M, Verfaillie G, Jouhaud F, Lachiver JM, Bataille N. How to manage the new generation of agile earth observation satellites. In: Proceeding of the international symposium on artificial intelligence, robotics and automation in space (ISAIRAS'00); 2000.

[28] Lemaître, M, Verfaillie G, Jouhaud F, Lachiver JM, Bataille N. Selecting and scheduling observations of agile satellites. Aerosp Sci Technol 2002; 6(5): 367-381.

[29] Li J, Li J, Chen H, Jing N. A data transmission scheduling algorithm for rapid-response earth-observing operations. Chinese J Aeronaut 2014; doi: http://dx.doi.org/10.1016/j.cja.2014.02.014.

[30] Li Y, Wang R, Xu M. Rescheduling of observing spacecraft using fuzzy neural network and ant colony algorithm. Chinese J Aeronaut 2014; doi: http://dx.doi.org/10.1016/j.cja.2014.04.027.

[31] Liao D, Tang Y. Satellite imaging order scheduling with stochastic weather condition forecast. In: Proceeding of the IEEE international conference on systems, man and cybernetics; 2005. 3: 2524-2529.

[32] Liao D, Tang Y. Imaging order scheduling of an earth observation satellite. IEEE Trans Syst Man Cy C 2007; 37(5): 794-802.

[33] Lin W, Liu C, Liao D, Lee Y. Daily imaging scheduling of an earth observation satellite. In: Prodeeding of the IEEE international conference on systems, man and cybernetics; 2003. 2: 1886 - 1891.

[34] Lin W, Liao D. A tabu search algorithm for satellite imaging scheduling. In: Prodeeding of the IEEE international conference on systems, man and cybernetics; 2004. 2: 1601-1606.

[35] Lin W, Chang S. Hybrid algorithms for satellite imaging scheduling. In: Proceeding of the IEEE international conference on systems, man and cybernetics; 2005. 3: 2518-2523.

[36] Lin W, Liao D, Liu C, Lee Y. Daily Imaging Scheduling of an Earth Observation Satellite. IEEE Trans Syst Man Cy A 2005; 35(2): 213-223.

[37] Luedtke J, Ahmed S. A sample approximation approach for optimization with probabilistic constraints. SIAM J Optimiz 2008; 19(2): 674-699.

[38] Luedtke J, Ahmed S, Nemhauser GL. An integer programming approach for linear programs with probabilistic constraints. Math Program 2010; 122(2): 247-272.

[39] Marinelli F, Salvatore N, Rossi F, Smriglio S. A lagrangian heuristic for satellite range scheduling with resource constraints. Comput Oper Res 2011; 38(11): 1572-1583.

[40] Padberg M, Rinaldi G. A branch-and-cut algorithm for the resource of large-scale symmetric traveling salesman problems. SIAM Rev 1991; 33(1): 60-100.

[41] Pemberton JC, Greenwald LG. On the need for dynamic scheduling of imaging satellites. In: Procding of the American society for photographing and remote sensing (ASPRS'02); 2002.

[42] Ruszczyński A. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. Math Program 2002; 93(2): 195-215.

[43] Sarkheyli A, Vaghei BG, Bagheri A. New tabu search heuristic in scheduling earth observation satellites. In: Proceeding of the second international conference on software technology and engineering (ICSTE'10); 2010.

[44] Soma P, Venkateswarlu S, Santhalakshmi S, Bagchi T, Kumar S. Multi-satellite scheduling using genetic algorithms. In: Proceeding of the eighth international conference on space operations (SpaceOps'04); 2004.

[45] Tangpattanakul P, Jozefowiez N, Lopez P. Multi-objective optimization for selecting and scheduling observations. In: Coello CA, Cutello V, Deb K, Forrest S, Nicosia G, Pavone M, editors. Lecture Notes in Computer Science, Springer; 2012.

[46] Vasquez M, Hao J. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. Comput Optim Appl 2001; 20(2): 137-157.

[47] Vasquez M, Hao J. Upper bounds for the SPOT 5 daily photograph scheduling problem. J Comb Optim 2003; 7(1): 87-103.

[48] Verfaillie G, Schiex T. Solution reuse in dynamic constraint satisfaction problems. In: Proceeding of the twelfth national conference on artificial intelligence (AAAI'94); 1994. p.307-312.

[49] Verfaillie G, Lemaitre M, Schiex T. Russian doll search for solving constraint optimization problems. In: Proceeding of the thirteenth national conference on Artificial intelligence (AAAI'96); 1996. p.181-187.

[50] Wang H, Xu M, Wang R, Li Y. Scheduling earth observing satellites with hybrid ant colony optimization algorithm. In: Proceeding of the international conference on artificial intelligence and computational intelligence (AICI'09); 2009.

[51] Wang J, Zhu X, Qiu D, Yang LT. Dynamic scheduling for emergency tasks on distributed imaging satellites with task merging. IEEE Trans Parall Distr 2014; 25(9): 2275-2285.

[52] Wang J, Zhu X, Yang LT, Zhu J, Ma M. Towards dynamic real-time scheduling for multiple earth observation satellites. J Comput Syst Sci 2014; doi: 10.1016/j.jcss.2014.06.016

[53] Wang P, Reinelt G, Gao P, Tan Y. A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation. Comput Ind Eng 2011; 61(2): 322-335.

[54] Williams DN. Time indexed formulation of scheduling problems (master thesis). Canada: Faculty of Commerce and Business Administration, University of British Columbia; 1997.

[55] Wolfe J, Stephen SE. Three scheduling algorithms applied to the earth observing systems domain. Manag Sci 2000; 46(1): 148-168.

[56] Wu G, Liu J, Ma M, Qiu D. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Comput Oper Res 2013; 40(7): 1884-1894.

[57] Zufferey N, Amstutz P, Giaccari P. Graph colouring approaches for a satellite range scheduling problem. J Scheduling 2008; 11(4): 263-277.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be